

HEWLETT-PACKARD COMPANY  
Intellectual Property Administration  
P.O. Box 272400  
Fort Collins, Colorado 80527-2400



PATENT APPLICATION

ATTORNEY DOCKET NO. 100110992-1

IN THE  
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): John Joseph MAZZITELLI

Confirmation No.: 1932

Application No.: 10/057,135

Examiner: Serrao, Ranodhi N.

Filing Date: October 29, 2001

Group Art Unit: 2141

Title: MULTI-THREADED SERVER ACCEPT SYSTEM AND METHOD

Mail Stop Appeal Brief-Patents  
Commissioner For Patents  
PO Box 1450  
Alexandria, VA 22313-1450

01/29/2007 CNEGA1 00000061 002025 10057135  
01 FC:1402 500.00 DA

TRANSMITTAL OF APPEAL BRIEF

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on November 30, 2006.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

☐ (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

☐ 1st Month  
\$120

☐ 2nd Month  
\$450

☐ 3rd Month  
\$1020

☐ 4th Month  
\$1590

☐ The extension fee has already been filed in this application.

☒ (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of \$ 500. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees.

☒ A duplicate copy of this transmittal letter is enclosed.

☒ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:  
Commissioner for Patents, Alexandria, VA 22313-1450  
Date of Deposit: January 25, 2007

OR

☐ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571)273-8300.

Date of facsimile:

Typed Name: Allen B. Kroger

Signature: [Signature]

Respectfully submitted,

John Joseph MAZZITELLI

By [Signature]  
James L. Baudino

Attorney/Agent for Applicant(s)

Reg No. : 43,486

Date : January 25, 2007

Telephone : (214) 855-7544



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

**APPEAL FROM THE EXAMINER TO THE BOARD  
OF PATENT APPEALS AND INTERFERENCES**

In re Application of: John Joseph MAZZITELLI Confirmation No.: 1932  
Serial No.: 10/057,135  
Filing Date: October 29, 2001  
Group Art Unit: 2141  
Examiner: Serrao, Ranodhi N.  
Title: MULTI-THREADED SERVER ACCEPT SYSTEM AND METHOD  
Docket No.: 100110992-1

**MAIL STOP: APPEAL BRIEF PATENTS**  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, Virginia 22313-1450

Dear Sir:

**APPEAL BRIEF**

Applicant has appealed to the Board of Patent Appeals and Interferences from the decision of the Examiner mailed August 30, 2006, finally rejecting Claims 1-6, 8-16, 18-26 and 28-30. Applicant filed a Notice of Appeal on November 30, 2006. Applicant respectfully submits herewith this Appeal Brief with authorization to charge the statutory fee of \$500.00.

### **REAL PARTY IN INTEREST**

The present application was assigned to Hewlett-Packard Company as indicated by an assignment from the inventor recorded on March 12, 2002 in the Assignment Records of the United States Patent and Trademark Office at Reel 012710, Frame 0191. The present application was subsequently assigned to Hewlett-Packard Development Company, L.P. as indicated by an assignment from Hewlett-Packard Company recorded on September 26, 2003 in the Assignment Records of the United States Patent and Trademark Office at Reel 014061, Frame 0492. The real party in interest is Hewlett-Packard Development Company, LP, a limited partnership established under the laws of the State of Texas and having a principal place of business at 20555 S.H. 249 Houston, TX 77070, U.S.A. (hereinafter "HPDC"). HPDC is a Texas limited partnership and is a wholly-owned affiliate of Hewlett-Packard Company, a Delaware Corporation, headquartered in Palo Alto, CA. The general or managing partner of HPDC is HPQ Holdings, LLC.

### **RELATED APPEALS AND INTERFERENCES**

There are no known appeals or interferences that will directly affect or be directly affected by or have a bearing on the Board's decision in this pending appeal.

### **STATUS OF CLAIMS**

Claims 1-6, 8-16, 18-26 and 28-30 stand rejected to a final Office Action mailed August 30, 2006. Claims 7, 17 and 27 have been cancelled without prejudice or disclaimer. Claims 1-6, 8-16, 18-26 and 28-30 are presented for appeal.

### **STATUS OF AMENDMENTS**

No amendment has been filed subsequent to the mailing of the Final Office Action.

### **SUMMARY OF CLAIMED SUBJECT MATTER**

Embodiments of the present invention as defined by independent Claim 1 are directed toward A multi-threaded server accept method comprising: creating a socket accept thread (31b) by a control thread (31a) of a server process (31); receiving a service request from a client (40a-40n) by the socket accept thread (31b); transferring the request to a data structure (35); retrieving the request, by the control thread (31a), from the data structure (35); and transferring the request to a client thread (31c-31m) dynamically created by the control thread (31a) to

process request data associated with the request (at least at page 4, lines 5-28; page 5, lines 6-33; page 6, lines 7-27; page 8, lines 10-33; page 9, lines 15-31; and figures 1 and 2A-2C).

Embodiments of the present invention as defined by independent Claim 12 are directed toward a multi-thread server accept system (10) comprising a server process (31) residing on a server (30) and operable to: create a socket accept thread (31b) by a control thread (31a) of the server process (31) residing on the server (30); receive a service request from a client (40a-40n) by the socket accept thread (31b); transfer the request to a data structure (35); retrieve the request, by the control thread (31a), from the data structure (35); and transfer the request to a client thread (31c-31m) dynamically created by the control thread (31a) to process request data associated with the request (at least at page 4, lines 5-28; page 5, lines 6-33; page 6, lines 7-27; page 8, lines 10-33; page 9, lines 15-31; and figures 1 and 2A-2C).

Embodiments of the present invention as defined by independent Claim 22 are directed toward a multi-threaded server accept application (31) comprising an application software residing on a computer-readable medium and operable to: create a socket accept thread (31b) by a control thread (31a) of the application software; receive a request from a client (40a-40n) by the socket accept thread (31b); transfer the request to a data structure (35); retrieve the request, by the control thread (31a), from the data structure (35); and transfer the request to a client thread (31c-31m) dynamically created by the control thread (31a) to process request data associated with the request (at least at page 4, lines 5-28; page 5, lines 6-33; page 6, lines 7-27; page 8, lines 10-33; page 9, lines 15-31; and figures 1 and 2A-2C).

#### **GROUND OF REJECTION TO BE REVIEWED ON APPEAL**

1. Claims 1-6, 8-16, 18-26 and 28-30 are rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,427,161 issued to LiVecchi (hereinafter "*LiVecchi*") and U.S. Patent Publication No. 2003/0088609 issued to Guedalia et al. (hereinafter "*Guedalia*").

#### **ARGUMENT**

A. Standard

1 35 U.S.C. § 103

To establish a *prima facie* case of obviousness under 35 U.S.C. § 103, three basic criteria must be met: First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art,

to modify the reference or to combine reference teachings; second, there must be a reasonable expectation of success; and finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Vaeck*, 947 F.2d 488, (Fed. Cir. 1991); M.P.E.P. § 2143. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. *Id.* Further, the mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680 (Fed. Cir. 1990); M.P.E.P. § 2143.01. Additionally, not only must there be a suggestion to combine the functional or operational aspects of the combined references, but also the prior art is required to suggest both the combination of elements and the structure resulting from the combination. *Stiftung v. Renishaw PLC*, 945 F.2d 1173, 1183 (Fed. Cir. 1991). Moreover, where there is no apparent disadvantage present in a particular prior art reference, then generally there can be no motivation to combine the teaching of another reference with the particular prior art reference. *Winner Int'l Royalty Corp. v. Wang*, 202 F.3d 1340, 1349 (Fed. Cir. 2000).

B. Argument

1. Rejection under 35 U.S.C. §103 in view of *LiVecchi* and *Guedalia*

Claims 1-6, 8-16, 18-26 and 28-30 are rejected under 35 U.S.C. §103(a) as being unpatentable over *LiVecchi* in view of *Guedalia*. Of the rejected claims, Claim 1, 12 and 22 are independent. Appellant respectfully submits that Claims 1, 12 and 22 are patentable over the cited references and, therefore, Claims 1, 12 and 22, and Claims 2-6, 8-11, 13-16, 18-21, 23-26 and 28-30 that depend respectively therefrom, are patentable.

In the final Office Action, the Appellee states that *LiVecchi* fails to teach "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited, for example, by Claim 1 (final Office Action, page 3). Appellant agrees. The Appellee further states that *Guedalia* purportedly teaches the above-referenced limitations of Claim 1, and that it would have been obvious to modify *LiVecchi* with the purported teaching of *Guedalia* to arrive at Appellant's invention as defined by Claims 1, 12 and 22 (final Office Action, page 3 ). Appellant respectfully disagrees.

Appellant respectfully submits that there is no motivation or suggestion to combine the purported teaching of *Guedalia* with *LiVecchi* as proposed by the Appellee and, further, at least *LiVecchi* teaches away from the proposed combination. For example, *LiVecchi* appears to

disclose two groups of worker threads for processing data requests: 1) active threads; and 2) blocked threads (comprised of worker threads not in the first or active group) (*LiVecchi*, column 7, lines 20-42). *LiVecchi* also appears to disclose a "novel scheduling heuristic" to determine whether to unblock a waiting thread to process a data request or wait for a currently-running thread to complete (*LiVecchi*, column 12, line 66 to column 13, line 2). Thus, in *LiVecchi*, there appears to be a finite number of "worker threads" for processing a data request. For example, *LiVecchi* recites the following mathematical expression to balance a finite number of worker threads with incoming workload:

$$R = (C * T * D) - (T/2)$$

(*LiVecchi*, column 13, line 13). Additionally, *LiVecchi* recites:

The objective of this scheduling heuristic is to balance the number of worker threads against the current incoming workload. Optimum results are achieved when over-scheduling does not occur. To do this, a small backlog should be maintained on the incoming ready queue (that is, some connections should be allowed to remain on the queue, and not be immediately assigned by awakening a worker thread).

(*LiVecchi*, column 13, lines 15-22) (emphasis added). *LiVecchi* further recites:

If the number of connections is greater than or equal to R when the test at Step 205 is made, then too many connections are already waiting . . . so a waiting thread will be unblocked by transferring control to Step 210. If the queue depth is less than R, then all the connections remain on the queue, waiting for running threads to finish.

(*LiVecchi*, column 13, lines 38-45) (emphasis added). *LiVecchi* also recites:

In other words, a system fitting the parameters of this example, can handle up to  $(C * T * D) - (T/2) = 11$  new requests every  $D = 0.2$  seconds by waiting for running threads to complete and assigning the new requests to those  $T = 10$  threads as they become available – and the requests do not have to wait on the queue longer than an average of  $D = 0.2$  seconds. If more than 11 new requests arrive, on average, during the period  $D = 0.2$  seconds, then the capacity of the system is exceeded and a waiting thread must be unblocked to ensure that the delay for any given request does not exceed D.

(*LiVecchi*, column 14, lines 16-26) (emphasis added). Thus, *LiVecchi* appears to use a novel scheduling heuristic to balance incoming workload by maintaining a certain number of worker

threads in queue and allowing them to remain on the queue for a certain period of time before unblocking a worker thread. Thus, Appellant respectfully submits that there is no motivation or suggestion to modify *LiVecchi* as proposed by the Examiner at least because the Appellee's proposed modification of *LiVecchi* would render *LiVecchi*'s "novel scheduling heuristic" inoperable for its intended purpose, namely, to prevent over-scheduling. In fact, to modify *LiVecchi* as proposed by the Appellee to "dynamically create" a thread to process request data would appear to completely ignore the explicit teaching of *LiVecchi* to maintain connections in a ready queue for a particular period of time and, after expiration of the time period, unblock an existing worker thread. Thus, there is clearly no motivation or suggestion to dynamically create a thread in *LiVecchi* as proposed by the Appellee as least because *LiVecchi* appears to clearly disclose that a finite quantity of threads is preferably used in a balanced manner to handle incoming work.

Moreover, *LiVecchi* teaches away from the modification proposed by the Appellee at least because *LiVecchi* expressly maintains connections in queue and waits for running threads to finish before assigning a thread to a request in order to prevent over-scheduling. *LiVecchi* clearly teaches away from dynamically creating threads to process incoming requests. Thus, at least *LiVecchi* teaches away from "dynamically creat[ing]" a client thread to process request data as recited by Claim 1. Accordingly, for at least these reasons, Applicant respectfully submits that independent Claim 1 is patentable over the *LiVecchi* and *Guedalia* references.

In the final Office Action, the Appellee asserts that *LiVecchi* does not teach away from Appellant's claimed invention because the Appellee asserts that both Appellant's invention and *LiVecchi* are directed toward enhancing the performance of a computer running a multi-threaded server application (final Office Action, page 2). However, this statement clearly fails to take into account what is actually taught by the *LiVecchi* reference, which Appellant respectfully submits clearly teaches away from "dynamically creat[ing]" a client thread to process request data as recited by Claim 1. Further, in the final Office Action, the Appellee appears to focus on a dividing a pool of worker threads by a dynamic partitioning process (final Office Action, page 2). However, dynamically partitioning a pool of threads is clearly not the same as "dynamically creat[ing]" a client thread. Thus, for at least these reasons, Appellant respectfully submits that Claims 1 is patentable over the cited references.

Independent Claim 12 recites "a server and operable to . . . transfer the request to a client thread dynamically created by the control thread to process request data associated with

the request" (emphasis added), and independent Claim 22 recites "an application software residing on a computer-readable medium and operable to . . . transfer the request to a client thread dynamically created by the control thread to process request data associated with the request" (emphasis added). At least for the reasons discussed above in connection with independent Claim 1, Applicant respectfully submits that independent Claims 12 and 22 are also patentable over the *LiVecchi* and *Guedalia* references.

Claims 2-6, 8-11, 13-16, 18-21, 23-26 and 28-30 depend respectively from independent Claims 1, 12 and 22. As discussed above, independent Claims 1, 12 and 22 are patentable over the *LiVecchi* and *Guedalia* references. Therefore, Claims 2-6, 8-11, 13-16, 18-21, 23-26 and 28-30 that depend respectively therefrom are also patentable. Accordingly, Appellant respectfully requests that Claims 1-6, 8-16, 18-26 and 28-30 be allowed.

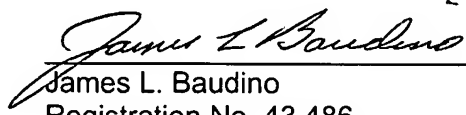


### CONCLUSION

Appellant has demonstrated that the present invention as claimed is clearly distinguishable over the art cited of record. Therefore, Appellant respectfully requests the Board of Patent Appeals and Interferences to reverse the final rejection of the Examiner and instruct the Examiner to issue a notice of allowance of all claims.

The Commissioner is authorized to charge the statutory fee of \$500.00 to Deposit Account No. 08-2025 of Hewlett-Packard Company. Although no other fee is believed due, the Commissioner is hereby authorized to charge any fees or credit any overpayments to Deposit Account No. 08-2025 of Hewlett-Packard Company.

Respectfully submitted,

  
\_\_\_\_\_  
James L. Baudino  
Registration No. 43,486

Date: January 25, 2007

Correspondence To:

Hewlett-Packard Company  
Intellectual Property Administration  
P.O. Box 272400  
Fort Collins, Colorado 80527-2400  
Tel. (970) 898-3884

## **CLAIMS APPENDIX**

1. A multi-threaded server accept method, comprising:  
creating a socket accept thread by a control thread of a server process;  
receiving a service request from a client by the socket accept thread;  
transferring the request to a data structure;  
retrieving the request, by the control thread, from the data structure; and  
transferring the request to a client thread dynamically created by the control thread to process request data associated with the request.
2. The method of claim 1, wherein the data structure comprises a queue.
3. The method of claim 1, wherein the data structure comprises a FIFO queue.
4. The method of claim 1, further comprising waiting for service requests by performing an accept ( ) call.
5. The method of claim 1, wherein receiving the request comprises receiving a client socket object.
6. The method of claim 1, further comprising waiting for the service request from the client by the socket accept thread.
8. The method of claim 1, further comprising:  
receiving a second request by the socket accept thread from the client;  
transferring the second request to the data structure;  
retrieving the second request by the control thread;  
transferring the second request to a second client thread to process second request data; and  
processing the second request data by the second client thread.

9. The method of claim 8, further comprising creating the second client thread to process the second request data.

10. The method of claim 1, wherein the socket accept thread and the control thread are executed on a single processor.

11. The method of claim 1, wherein the steps of transferring the request to the data structure and retrieving the request from the data structure are serially performed.

12. A multi-thread server accept system, comprising:

a server process residing on a server and operable to

create a socket accept thread by a control thread of a server process residing on the server;

receive a service request from a client by the socket accept thread;

transfer the request to a data structure;

retrieve the request, by the control thread, from the data structure;

transfer the request to a client thread dynamically created by the control thread to process request data associated with the request.

13. The system of claim 12, wherein the data structure comprises a queue.

14. The system of claim 12, wherein the data structure comprises a FIFO queue.

15. The system of claim 12, wherein the socket accept thread is operable to wait for service requests by performing an accept( ) call.

16. The system of claim 12, wherein the socket accept thread is operable to receive the request by receiving a client socket object from the client.

18. The system of claim 12, wherein the server process is further operable to:  
receive a second request from the client by the socket accept thread after transferring the request to the data structure;  
transfer the second request to the data structure;  
retrieve the second request by the control thread;  
transfer the second request to a second client thread to process the second request data; and  
process the second request data by the second client thread.

19. The system of claim 18, wherein the server process is further operable to create the second client thread to process the second request data.

20. The system of claim 12, wherein the socket accept thread and the control thread are executed on a single processor.

21. The system of claim 12, wherein the server process is further operable to serially perform the steps of transferring the request to the data structure and retrieving the request from the data structure.

22. A multi-threaded server accept application, comprising:

an application software residing on a computer-readable medium and operable to:

create a socket accept thread by a control thread of the application software;  
receive a request from a client by the socket accept thread;  
transfer the request to a data structure;  
retrieve the request, by the control thread, from the data structure;  
transfer the request to a client thread dynamically created by the control thread to process request data associated with the request.

23. The application of claim 22, wherein the data structure comprises a queue.

24. The application of claim 22, wherein the data structure comprises a FIFO queue.

25. The application of claim 22, wherein the application software is further operable to wait for service requests by calling an accept( ) program.

26. The application of claim 22, wherein the application is further operable to receive the request by receiving a client socket object from the client.

28. The application of claim 22, wherein the application software is further operable to:  
receive a second request from the client by the socket accept thread after transferring the request to the data structure;

transfer the second request to the data structure;

retrieve the second request by the control thread;

transfer the second request to a second client thread to process second request data;

and

process the second request data by the second client thread.

29. The application of claim 22, wherein the socket accept thread and the control thread are executed on a single processor.

30. The application of claim 22, wherein the application software is further operable to serially perform the steps of transferring the request to the data structure and retrieving the request from the data structure.

## **EVIDENCE APPENDIX**

None

**RELATED PROCEEDINGS APPENDIX**

None